Ministry of Education of Moldova

Technical University of Moldova

Faculty "Computers, Informatics and Microelectronics"

Report

Laboratory work nr. 1
on Embedded systems

Performed by:
st. gr. FAF-141 D. Lupei

Verified by:
assoc. prof., A. Bragarenco

Chisinau -2016

Topic

Introduction to Micro Controller Unit programming. Implementing serial communication over UART – Universal asynchronous receiver/transmitter.

Objectives

- Basic knowledge about Micro Controller Unit programming
- Implementing and learn about serial communication over UART

Tasks

Write a C program and design schematics for Micro Controller Unit (MCU) using Universal asynchronous receiver/transmitter. For writing program use AVR Compiler and for schematics use Proteus, which allow us to simulate real example.

Used Tools

Atmel Studio is the integrated development platform (IDP) for developing and debugging Atmel® SMART ARM®-based and Atmel AVR® microcontroller (MCU) applications. Studio 7 supports all AVR and Atmel SMART MCUs. The Atmel Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. It also connects seamlessly to Atmel debuggers and development kits.

Proteus developed by Labcenter Electronics, is a software with which you can easily generate schematic captures, develop PCB and simulate microprocessor. It has such a simple yet effective interface that it simplifies the task required to be performed. This one aspect has attracted many users to select this tool amongst many others offering the same services.

Short theory

A microcontroller (sometimes abbreviated µC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

Some microcontrollers may use four-bit words and operate at frequencies as low as 4 kHz, for low power consumption (single-digit milliwatts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

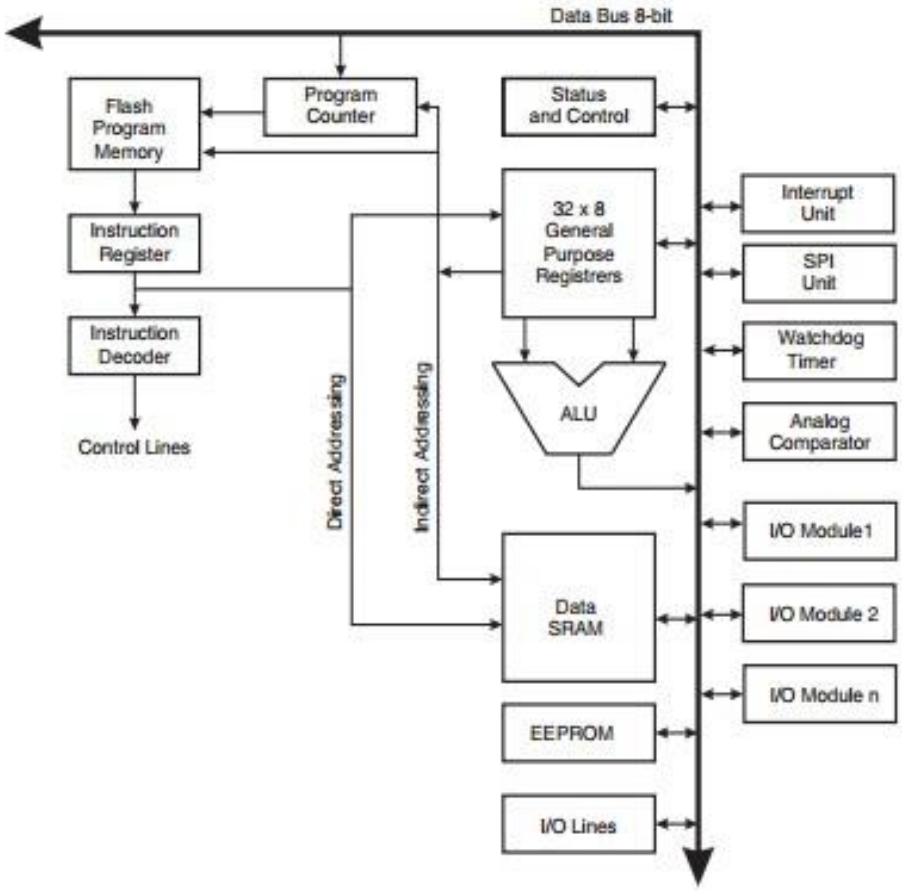


Fig. 1.General MCU Diagram

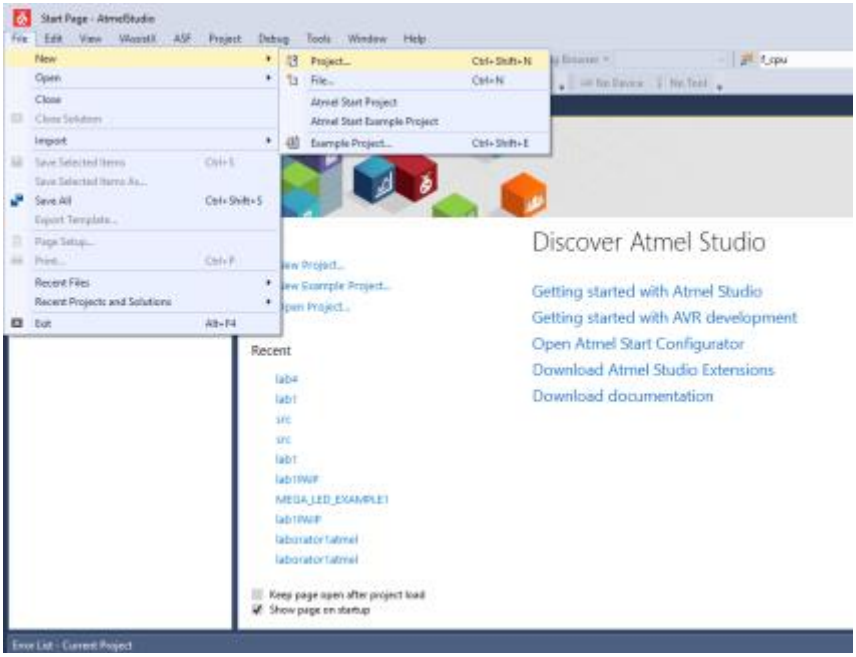
A UART (Universal Asynchronous Receiver/Transmitter) is the microchip with programming that controls a computer's interface to its attached serial devices. Specifically, it provides the computer with the RS-232C Data Terminal Equipment (DTE) interface so that it can "talk" to

and exchange data with modems and other serial devices. As part of this interface, the UART also:

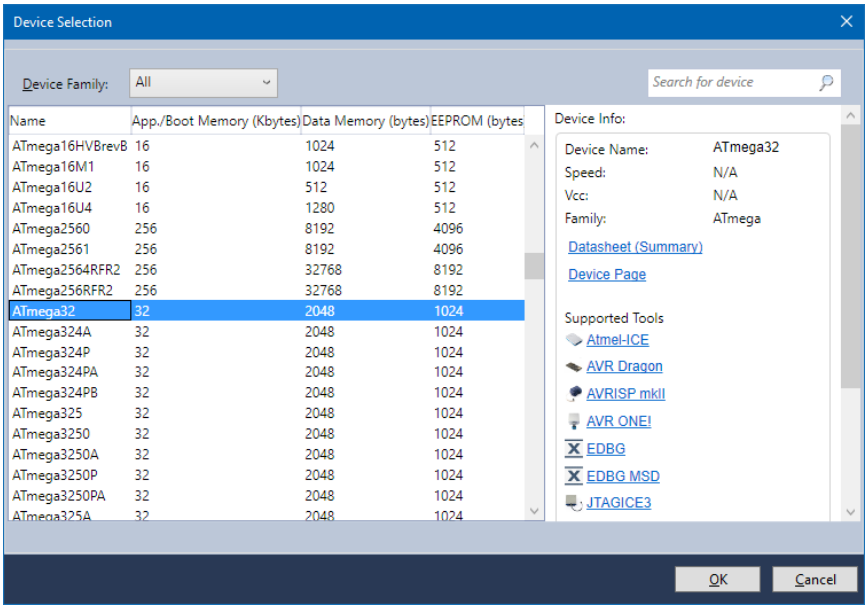
- Converts the bytes it receives from the computer along parallel circuits into a single serial bit stream for outbound transmission
- On inbound transmission, converts the serial bit stream into the bytes that the computer handles
- Adds a parity bit (if it's been selected) on outbound transmissions and checks the parity of incoming bytes (if selected) and discards the parity bit
- Adds start and stop delineators on outbound and strips them from inbound transmissions
- Handles interrupt s from the keyboard and mouse (which are serial devices with special port s)
- May handle other kinds of interrupt and device management that require coordinating the computer's speed of operation with device speeds

Work process:

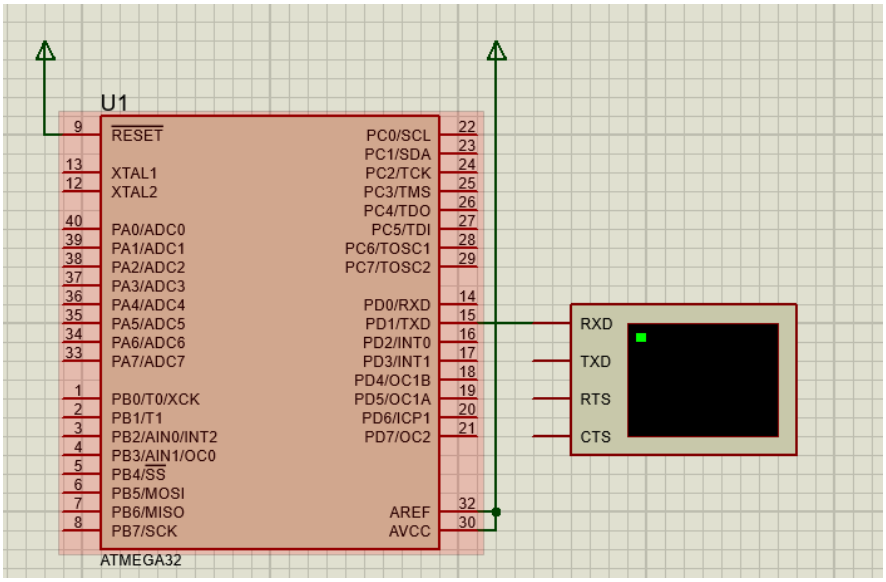
1. To start we should familiarize with the Atmel Studio IDE. After launching the program it’s opens the main window where we select File/Project for creating a new project



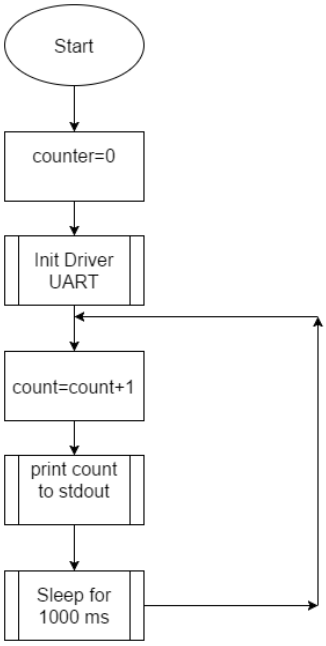
2. Then appear windows from which we should the needed microcontroller in our case Atmega32.



3. First steps is to formulate the principle of operation of the microcontroller. The next step is to create the circuit with the simulation software Proteus where we need only a ATmega32 MCU and a peripheral UART device.

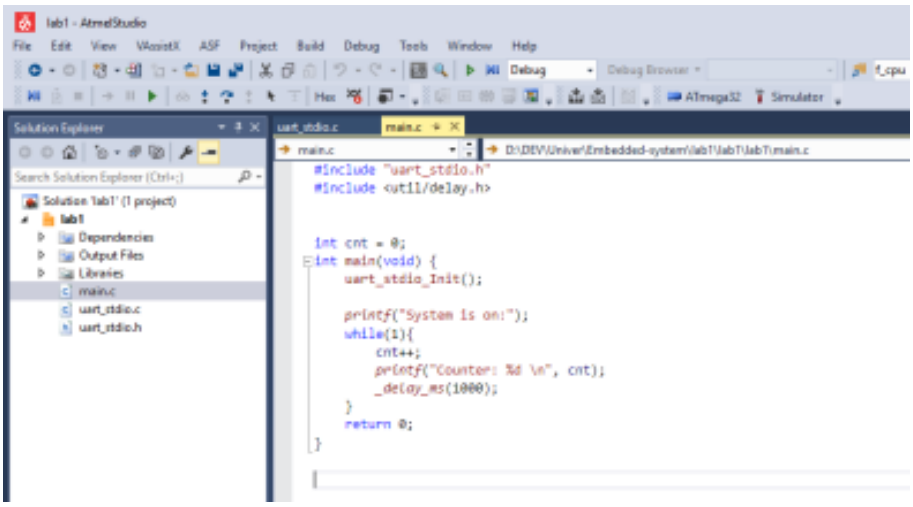


4 . After designing the scheme we have to implement the program itself and to write the code according to the draw flowchart.

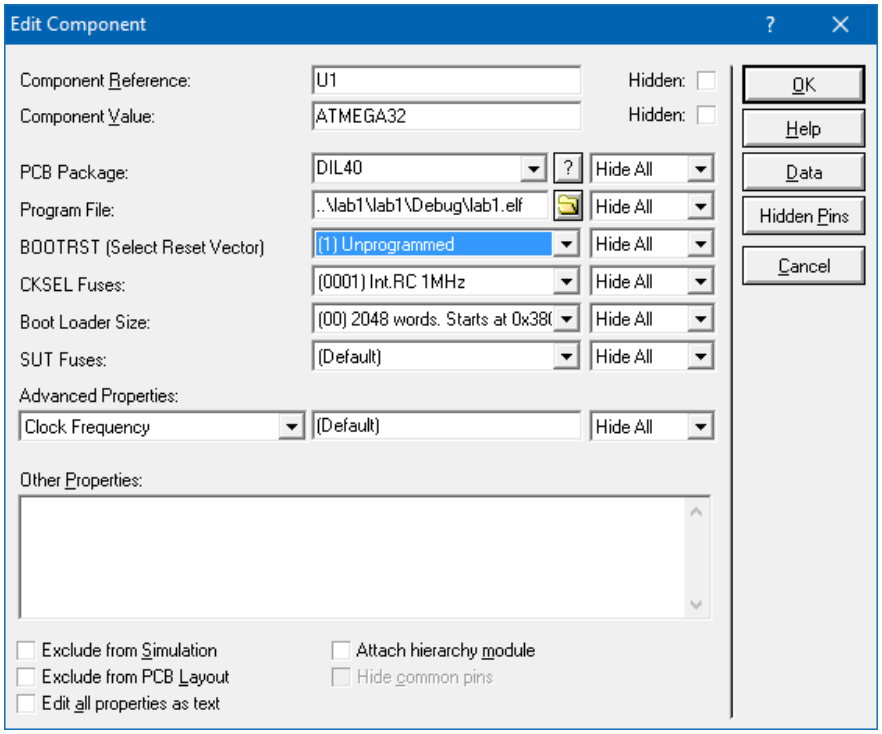


5.The program itself is not very hard to understand and to write . In main.c file we have a void function where we declare variable for counting then we intialize UART driver and start infinite loop. T

6. The trickest part of this program is that in order to use UART we have to write a Driver which will interact with the peripheral device. After code implementation, we should now **Build Hex and Elf** which will be written to MCU ROM. The general project structure look like in the following image:



7. After compiling and verifying code for errors and builiding solution we need to make sure that the MCU is connected to Virtual Terminal. We should connect our microcontroller and make sure that Tx is connected to Peripheral Rx. The next step is to attach program to our virtual MCU. This is done trough ELF or HEX file which we put in Edit Component/Program File in Proteus.



The last step is simulating the circuit in PROTEUS, after which we can be convinced that the program works okay and microcontroller executes all commands accurately, therefore obtaining the desired result.

Conclusion: After doing this laboratory work we obtain basic knowledge about architecture of microcontrollers , its basic work process, MCU Programming and designing schemes and simulate microprocessors. Also I learned that nowdays a lot of tech things starting from computers to cars, planes , robots and automatically controlled products and devices contains microcontrollers and that embed systems is an area that will have a huge growth over the next years.

Bibliography

- M.A. Mazidi, "*AVR microcontroller and embedded system using C and assembly language*", 2009
- Thomas Bräunl, "*Embedded Robotics*"
- **http://www.microlab.club/search/**label/Laborator

Appendix

main.c

```
#include "uart_stdio.h"
#include <util/delay.h>


int cnt = 0;
int main(void) {
	uart_stdio_Init();

	printf("System is on:");
	while(1){
		cnt++;
		printf("Counter: %d \n", cnt);
		_delay_ms(1000);
	}
	return 0;
}
```

uart_stdio.c

```
#include "uart_stdio.h"

#define UART_BAUD  9600

#include <avr/io.h>
#include <stdio.h>

FILE std_out =  FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_RW);

int uart_putchar(char c, FILE *stream) {
	while(~UCSRA &(1<<UDRE)); //wait while register is free
	UDR = c; //load c in register

	return 0;
}

void uart_stdio_Init(void) {

	#if F_CPU < 2000000UL && defined(U2X)
	UCSRA = _BV(U2X);		/* improve baud rate error by using 2x clk */
	UBRRL = (F_CPU / (8UL * UART_BAUD)) - 1;
	#else
	UBRRL = (F_CPU / (16UL * UART_BAUD)) - 1;
	#endif
	UCSRB = _BV(TXEN) | _BV(RXEN); /* tx/rx enable */
	stdout = &std_out;
}
```

uart_stdio.h

```
#ifndef UART_STDIO_H_
#define UART_STDIO_H_

#define F_CPU 1000000UL
#include <stdio.h>

void uart_stdio_Init(void);
int uart_putchar(char c, FILE *stream);

#endif /* UART_STDIO_H_ */
```